Nicholas Chan
nipchan@ucsc.edu
5/13/2021
CSE13s Spring ‘21

Design Document:
Assignment 6: Huffman Coding

Huffman coding is a system for compressing data based on the concept of entropy. The entropy or measure of chaos within a file can be represented through the construction of a histogram covering all the permutations of a uint8_t (there will be 256 entries in the histogram/array). Based on the distribution of frequencies among uint8_t characters, less frequent uint8_t’s receive a code with more bits while more frequent uint8_t’s receive a code with less bits.

In assignment 6, I have been given the task of implementing an encoder, decoder and IO module for a huffman coding interface as well as ADTs for nodes, priority queues, stacks. Encoding is a process which will require the construction of a Huffman tree from nodes. A priority queue will be used to order the nodes before they are dequeued and added to the Huffman tree. Characters will be encoded based on their positions in the tree.

Top Level:

Overview

- Encoding Part:
    - Histogram
        - Array of 256 uint64_t items
        - Indices are characters, values frequencies
    - Node ADT
    - Stack (Uses Node ADT)
    - Priority Queue (Uses Node ADT)
    - Huffman Coding Module (Uses Priority Queue to build the Huffman Tree)
    - Code ADT
    - I/O
- Decoding Part (Reconstructing and traversing the Huffman Tree):
    - Huffman Coding Module (No Priority queue)
    - Stack
    - Node
    - I/O

Node ADT

- The node struct contains
    - Node pointer to left
    - Node pointer to right
    - uint8_t symbol
    - uint64_t frequency
- node_join takes in pointers to left and right nodes, returns a node pointe
    - Create new node named n
    - Symbol of n = $
    - Frequency of n = sum(left and right freqs)
    - Node to the left of n = left
    - Node to the right of n = right

Priority Queue PQ ADT (Similar to regular queue)

- PQ struct contains
    - uint32_t for Capacity
    - uint32_t for Size
    - uint32_t for Head
    - uint32_t for Tail
- Enqueue takes in a priority queue pointer q and a node pointer n to enqueue and returns a bool
    - Create a placeholder index called tmp
    - While tmp != head

        - If frequency of node pq[tmp-1] >frequency of node n
            - Copy node pq[tmp-1] over to node pq[tmp]
            - tmp -= 1
        - Else
            - Node pq[tmp] is set to node n

Code Module (All functions except init take a pointer)

- Code struct contains:
    - uint32_t for top
        - Used to get the top bit like in the stack struct
    - uint8_t array of byte sized items
        - Size of array = MAX_CODE_SIZE macro
        - The bitvector
- code_init is the constructor of Code instances
    - Create Code called c
    - Set top of c = 0
    - Return pointer to Code c
- code_size returns the size of a code given a Code pointer

I/O System Calls

- read(file descriptor, buffer, number of bytes) and write(file descriptor, buffer, number of bytes)
    - man read() and write()
    - Takes in buffer, number of bytes and file descriptor
    - Buffer size is specified as the BLOCK macro (4096 bytes)

References:

- https://www.youtube.com/watch?v=joX93VhNlRo